REMARKS

I. Introduction

Claims 17, 19-22, 24-28 and 30-35 are pending after the addition of claims 34 and 35. Claims 17, 27 and 32 have been amended. For at least the reasons set forth below, Applicants submit that the pending claims are in condition for allowance.

II. Claim Objection

Applicants have amended claims 17, 27 and 32 in order to obviate the Examiner's objection to claims 17, 19-22, 24-28 and 30-33. Withdrawal of the claim objection is respectfully requested.

III. Rejections under 35 U.S.C. § 103

Claims 17, 19-21 and 26-28 and 30-33 were rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,872,977 ("Thompson") in view of U.S. Patent Pub. No. 2003/0192036 ("Karkare").

In order for a claim to be rejected for obviousness under 35 U.S.C. § 103(a), the prior art must teach or suggest each element of the claim. See Northern Telecom, Inc. v. Datapoint Corp., 908 F.2d 931, 934 (Fed. Cir. 1990), cert. denied, 111 S. Ct. 296 (1990); In re Bond, 910 F.2d 831, 834 (Fed. Cir. 1990). To establish a *prima facie* case of obviousness, the Examiner must show, *inter alia*, that there is some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify or combine the references, and that, when so modified or combined, the prior art teaches or suggests all of the claim limitations. M.P.E.P. §2143. In addition, as clearly indicated by the Supreme Court, it is "important to identify a reason that would have prompted a person of ordinary skill in the relevant field to combine the [prior art] elements" in the manner claimed. See KSR Int'l Co. v. Teleflex, Inc., 127 S. Ct. 1727 (2007). In addition, all the teachings of the prior art must be considered, including those which teach away from the claimed invention. (See M.P.E.P. 2143.01.II). To the extent that the Examiner may be relying on the doctrine of inherent disclosure to support the obviousness rejection, the Examiner must provide a "basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristics necessarily flow from the

teachings of the applied art." (See M.P.E.P. § 2112; emphasis in original; see also Ex parte Levy, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990)).

Amended claim 17 recites, in relevant parts, the following:

at least one of:

creating at least one implementation-independent configuration data file, and

altering information filed in the at least one implementation-independent configuration data file;

wherein the information stored in **the at least one implementation-independent configuration data file describes concrete configuration values in an abstract fashion**;

using a computer script, at least one of automatically setting-up and automatically updating **configuration data, stored in a configuration data container, as a function of the information filed in the at least one implementation-independent configuration data file, wherein the configuration data are extracted from the implementation-independent configuration data file** and stored in the configuration data container;

automatically generating at least one item of dependency information describing a dependency on at least two configuration data present in the configuration data container;

automatically generating at least one implementation-dependent configuration data file as a function of the configuration data stored in the configuration data container, and as a function of the at least one item of dependency information, wherein **concrete configuration data are stored in the implementation-dependent configuration data file**; and

automatically configuring the at least one functional unit as a function of concrete information filed in the at least one implementation-dependent configuration data file.

Claims 27 and 32 have been amended to recite limitations substantially similar to the above-recited limitations of claim 17.

The claimed subject matter relates to a method for configuring a computer program including at least one functional unit, by creating/altering at least **one implementation-independent configuration data file**. Configuration data is then set-up or updated in a **configuration data container** as a function of the information that is stored in the implementation-independent configuration data file. The configuration data stored in the configuration data container is then used to generate at least **one implementation-dependent configuration data file**, and the implementation-dependent configuration data file is used to configure the at least one functional unit. Thus, the data that ultimately determines a specific configuration of the at least one functional unit is filed independently of the intended, specific implementation, i.e., filed as abstract data in the implementation-independent configuration data file(s).

The implementation-independent configuration data file describes concrete configuration values or configuration parameters in an abstract fashion. For instance, concrete configuration values may define the measuring range of a sensor module for measuring an electric voltage. Illustratively, it is possible to specify a measuring range abstractly with the values 3-5 volts. However, the implementation-dependent values of the measuring range to be generated therefrom may lie between 10,000 and 20,000, for example. The concrete and thus implementation-dependent values could then be for example 10,000 and 20,000 and might be used to generate a header file (e.g. "file_2.h" in Figure 1 of the present application) in order to further instantiate a variable used in a source file (e.g., "file_2.c" in Figure 1). These values might then be used in a functional unit corresponding to a sensor module. In this case, the functional unit would have to be configured using these concrete configuration values 10,000 and 20,000, for instance, to permit a measurement in a measuring range of 3-5 volts.

In contrast to the configuration method described above, **conventional methods use header files that already contain concrete configuration parameters**. Whenever the configuration has to be changed, the concrete values in the header files have to be changed and eventually the whole software project has to be compiled in order to generate machine code (e.g., executable binaries). Whenever the implementation has to be altered, e.g. because a sensor module is replaced by another, the relevant values in the header files have to be changed or updated, in case the new sensor module has to be addressed with different concrete values (in order to fulfill the same task as the former sensor module).

In order to simplify software creation and in order to reduce failures due to wrong configurations, **the claimed invention provides using an abstract configuration file for storing abstract configuration data** (e.g., the measuring range from 3 to 5 volts) **instead of concrete values. An implementation-dependent configuration data file containing concrete data is then generated based on the abstract data (since the implementation-dependent configuration data file is extracted from the abstract configuration data of the implementation-independent data file) and used to configure the functional unit.**

Thompson teaches a configuration method that involves using a "makefile", which contains instructions that are used to control the compilation and linking of source code modules (col. 2, lines 7 - 12). The makefile includes instructions containing targets, dependencies and commands for updating the targets (col. 2, lines 14 - 17). The target is an executable program (col. 2, lines 31 - 33). Thus, the makefiles of Thompson are used to drive the compilation process, i.e., to combine source files and object code into an executable program (col. 2, lines 33 - 48).

Thompson further discloses a platform-independent "build file" which is used to generate the makefile. This is achieved by the use of a "rules file" which is instantiated with arguments that are included in the build file and passed to the rules file (col. 2, lines 66 - col. 3, line 11).

Therefore, **(a) the build file of Thompson is analogous to a conventional header file**, since header files also comprise arguments that are used in order to instantiate variables that are included in source files, and since header files include code fragments that are copied into object files; **(b) the rules file of Thompson is analogous to a conventional source file** since both are instantiated by using arguments instead of place holders, e.g., variables; and **(c) the makefile of Thompson is analogous to a conventional object file or executable file**, since each makefile is an executable script used for driving the compilation process.

Based on the description above, it is apparent that Thompson's configuration method relates to a **compilation** procedure, whereas according to the present invention, **the creation of the implementation-independent configuration data files and implementation-dependent configuration data files takes place outside of compilation**. To illustrate this difference, the build file and rules file of Thompson may be considered as being a separate input to the compiler 6 in Figure 1 of the present application, with the additional step of

generating a makefile prior to compiling. Thus, although Thompson provides for a certain degree of **platform independency for driving a compilation process**, the present invention goes beyond the teachings of Thompson to achieve a certain degree of **implementation independency for generating source files that might be input to the compilation process.**

In sum, Thompson relates to a conventional compilation procedure and does not disclose or suggest **an implementation-independent configuration file comprising abstract configuration data**. Instead, **Thompson only discloses a platform independent file (e.g., the build file) that comprises arguments**. However, arguments cannot possibly be **abstract information**, since arguments are concrete information that are used to instantiate variables. **Similarly, Thompson's platform-dependent file (e.g., the rules file)** includes place holders, e.g. variables, which **are also not abstract information**, since they do not have any meaning until they are instantiated by concrete arguments.

Additionally, because Thompson does not disclose or suggest abstract configuration data, Thompson also fails to disclose or suggest: **a configuration data container for storing configuration as a function of the information filed in the implementation-independent configuration data file; and automatically generating at least one item of dependency information describing a dependency on at least two configuration data present in the configuration data container**. Instead, Thompson discloses the generation of **dependencies of the source files and the object files**.

Karkare fails to remedy the critical deficiencies of Thompson described above in connection with claim 17. Accordingly, claims 17, 27 and 32, as well as their dependent claims 19-21 and 26, 28, 30, 31 and 33, are not rendered obvious by the combination of Thompson and Karkare.

Claim 22 was rejected under 35 U.S.C. § 103(a) as unpatentable over Thompson in view of Karkare and further in view of U.S. Patent No. 7,234,135 ("Bollhoefer"). Claims 24 and 25 were rejected under 35 U.S.C. § 103(a) as unpatentable over Thompson in view of Karkare and further in view of U.S. Patent Pub. No. 2002/0040469 ("Pramberger").

Claims 22, 24 and 25 ultimately depend on claim 17. Bollhoefer and Pramberger fail to remedy the critical deficiencies of Thompson and Karkare described above in connection with claims 17. Accordingly, claim 17 and its dependent claims 22, 24 and 25, are not rendered obvious by the combination of Thompson, Karkare, Bollhoefer and Pramberger. Withdrawal of the obviousness rejections of claims 22, 24 and 25 is respectfully requested.

Conclusion

Applicants respectfully submit that all pending claims 17, 19-22, 24-28 and 30-35 of the present application are in condition for allowance. Prompt reconsideration and allowance of the present application are therefore earnestly solicited.

Respectfully submitted,

Dated: March 12, 2012

By: /Jong H. Lee/
Jong H. Lee (Reg. No. 36,197), for:

Gerard A. Messina (Reg. No. 35,952)
KENYON & KENYON LLP
One Broadway
New York, New York 10004
(212) 425-7200
CUSTOMER NO 26646